A Proofs for continuous contraction algorithms

We will first prove theorem 2 for s-t-mincuts. Afterwards, we will give those parts of the proof for the normalized cut version that differ from the version for s-t-mincuts.

Proof of theorem 2 for s-t-mincuts. Fix a value of n and a continuous contraction algorithm, we will then construct a graph on which this algorithm finds an s-t-mincut with probability $\leq 2^{-n+3}$.

Consider the graph G in fig. 1 in the main paper, but with weight 1 for all edges and generalized to n vertices (two of which are s and t). We call the adjacency matrix of this graph A. Every s-t-cut of G is minimal, and there are 2^{n-2} such s-t-cuts. So there must be at least one s-t-mincut, whose cut set we shall call C, that the contraction algorithm selects with probability $\leq 2^{-n+2}$.

The idea of this proof is to slightly decrease the weights of all the edges in C. Because of continuity, we can do this in such a way that the probability of selecting C does not change by much. Then the cut defined by C will be the unique s-t-mincut in the modified graph but will still be selected with probability of order 2^{-n}. What follows is a more rigorous version of this argument.

We will write $p_C(\tilde{A})$ for the probability that the algorithm selects the cut C on the graph with n vertices and weighted adjacency matrix $\tilde{A}$. We know that $p_C(A) \leq 2^{-n+2}$. Our goal is to find an adjacency matrix A' for which C is the unique s-t-mincut and $p_C(A') \leq 2^{-n+3}$.

First, we need to show that for a continuous contraction algorithm, $p_C(\tilde{A})$ is a continuous function of $\tilde{A}$. The definition of continuous contraction algorithms only states that the scores at each step need to be continuous function of the adjacency matrix. It's unsurprising that this also leads to continuous overall probabilities of selecting given cuts. The details do not provide much insight and are shown separately as lemma 1.

This continuity of $p_C(\tilde{A})$ means that for $\varepsilon := 2^{-n+2}$, there is a $\delta > 0$ such that if $\|A - A'\|_\infty \leq \delta$ for some A', then $|p_C(A) - p_C(A')| \leq \varepsilon$.

So we define the graph G' with adjacency matrix A' by setting the weights of the edges in the cut set C to $1 - \delta$ and leaving the other weights at 1[1]. Then $\|A - A'\|_\infty = \delta$, so

$$|p_C(A) - p_C(A')| \leq \varepsilon = 2^{-n+2}$$

This means that the probability of finding the cut C in the new graph G' is

$$p_C(A') \leq p_C(A) + \varepsilon \leq 2^{-n+2} + 2^{-n+2} = 2^{-n+3}$$

At the same time, C is the *unique* s-t-mincut of G'. Every s-t-cut has a cut set with the same cardinality and C is the only one which contains only edges that have weight $1 - \delta$ — every other cut set contains some edges with weight 1.

This means that on G', the contraction algorithm has only an exponentially low probability of finding *any* s-t-mincut, as claimed. □

[1] We can of course assume $\delta < 1$ without loss of generality

Proof of theorem 2 for normalized cuts. Instead of the graph from fig. 1 in the main paper that we used in the previous proof, let G be a complete unweighted graph on n vertices. We will show that in this graph, every cut is a normalized cut.

In a complete unweighted graph, we always have

$$w(A, B) = \sum_{a \in A, b \in B} w_{ab} = |A| \cdot |B| - |A \cap B|$$

(the last term is necessary because there are no self-loops). Therefore, with $k := |A|$, we get

$$\begin{aligned} w(A, A^c) &= k(n-k) \\ w(A, V) &= k(n-1) \\ w(A^c, V) &= (n-k)(n-1) \end{aligned}$$

So the normalized cut cost of the cut (A, A^c) is

$$\begin{aligned} \operatorname{ncut}(A, A^c) &= \frac{w(A, A^c)}{w(A, V)} + \frac{w(A, A^c)}{w(A^c, V)} = \frac{k(n-k)}{k(n-1)} + \frac{k(n-k)}{(n-k)(n-1)} \\ &= \frac{(n-k)+k}{n-1} = \frac{n}{n-1} \end{aligned}$$

for each of the 2^{n-1} possible cuts. This means that there are 2^{n-1} normalized cuts, and the algorithm must assign probability $\leq 2^{-n+1}$ to at least one of them.

From here on the proof procedes like that for s-t-mincuts: if the weights are slightly perturbed, there will be a unique normalized cut, but its probability will still be close to 2^{-n+1}. We therefore don't repeat the details. □

To make working with weighted adjacency matrices easier, we consider every graph to be fully connected for the following Lemma. Non-existent edges are instead treated as edges with weight zero.

Lemma 1. *Let C be a fixed set of edges of the complete graph on n vertices. Let $p_C(A)$ be the probability that a given continuous contraction algorithm does not contract any edges from C when run on the graph with n vertices and weighted adjacency matrix A. Then p_C is a continuous function.*

Note that we don't require C to be a cut set and that here, $p_C(A)$ does not always denote the probability that C is the chosen cut. This makes the proof more concise and is a strict generalization: if C happens to be a cut set for some adjacency matrix A, then $p_C(A)$ will be the probability that C is the final chosen cut.

Proof. Let $e_1, \ldots, e_{n-2}$ be an arbitrary but fixed set of edges. We will show that the probability $p(e_1, \ldots, e_{n-2})$ that the algorithm contracts $e_1, \ldots, e_{n-2}$ in that order is a continuous function of A. Then the claim follows because $p_C(A)$ is simply the sum of these probabilities over all $(n-2)$-tuples of edges that don't contain edges from C.

We prove the claim in two steps:

1. Let $C_k(A)$ for $k \in \{1, \ldots, n-2\}$ be the adjacency matrix that is reached from starting with A and contracting $e_1, \ldots, e_k$. We will show that $C_k(A)$ is a continuous function for all k.

2. We then show that $p(e_1, \ldots, e_{n-2})$ is a continuous function of the partially contracted adjacency matrices $C_1(A), \ldots, C_{n-2}(A)$

It will then follow that $p(e_1, \ldots, e_{n-2})$ and thus p_C is a continuous function of A, as a composition of continuous functions.

For the first step, note that contracting an edge sets an entry in the adjacency matrix to zero and adds its previous value to another entry. This means that each entry of $C_1(A)$ is either zero (for all A) or a certain sum of entries of A. The structure of the sum is determined by e_1 and does not depend on A. Therefore, C_1 is a continuous function of A. Each entry in $C_2(A)$ is zero or a sum of entries in $C_1(A)$ and therefore a continuous function of $C_1(A)$, which makes it a continuous function of A by composition. By induction it follows that all C_k are continuous functions of A.

For the second step, we write $p(e_1, \ldots, e_{n-2})$ as

$$p(e_1, \ldots, e_{n-2}) = \prod_{k=1}^{n-2} \frac{\mathcal{W}(e_k; C_k(A))}{\sum_e \mathcal{W}(e; C_k(A))} \tag{1}$$

the sum over e is over all the edges at that step; which summands appear depends only on $e_1, \ldots, e_{n-2}$ and not on A. The scores $\mathcal{W}$ that appear are by definition continuous in their second argument $C_k(A)$. Since $C_k(A)$ is continuous in A and the entire expression is clearly continuous in the scores, $p(e_1, \ldots, e_{n-2})$ is continuous in A as claimed. □

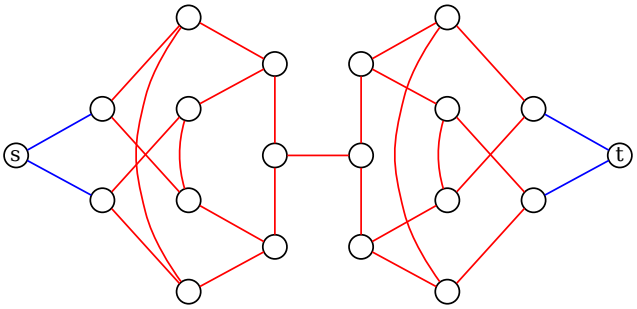
B Proofs for local contraction algorithms

We will first prove theorem 3 for s-t-mincuts. The approximability result from corollary 4 and the result for normalized cuts will then follow easily.

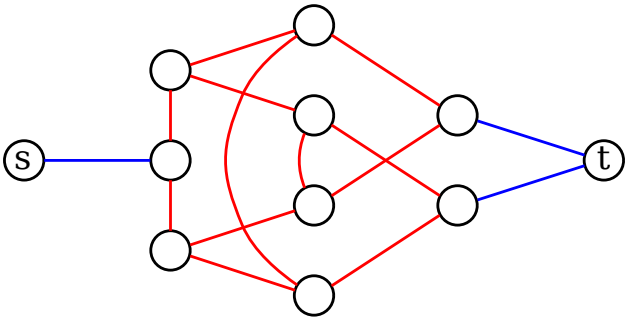
Proof of theorem 3 for s-t-mincuts. The graphs G_n we will use consist of n copies of each of three different subgraphs, shown in fig. 1. As an example, a schematic version of G_2 is shown in fig. 2. Each of the colored boxes contains one subgraph, each of the three types occurs twice. The different box colors denote the three different types. The last two types only differ in their orientation but we will treat them separately. The general graph G_n simply has n instead of two copies of each subgraph.

We call each such subgraph, including the blue edges that connect it to s and t, a *band*. There are $3n$ bands, n of each type. We say that a band has been *touched* if one of the edges belonging to it has been contracted. Clearly, every band has to be touched at some point during the contraction process.

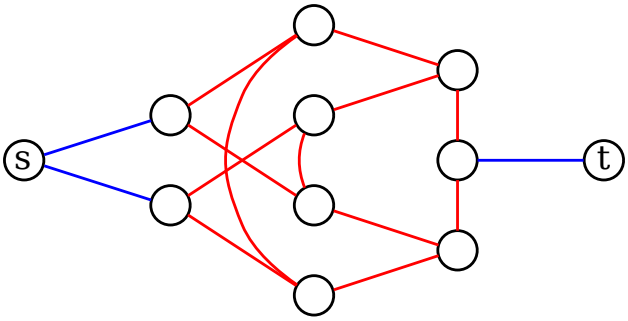
One of the key ideas of the proof is that G_n contains many edges that have isomorphic neighborhoods, but some of which are part of the s-t-mincut while



(a) Band of type A



(b) Band of type B



(c) Band of type C

Figure 1: The three different "bands" used in G_n

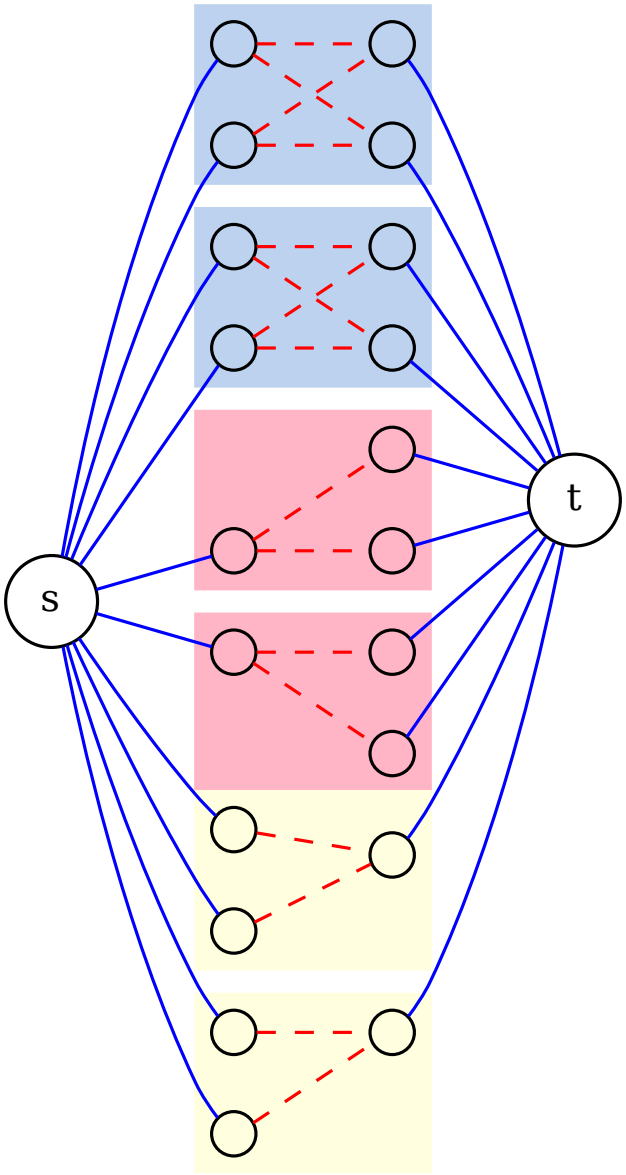
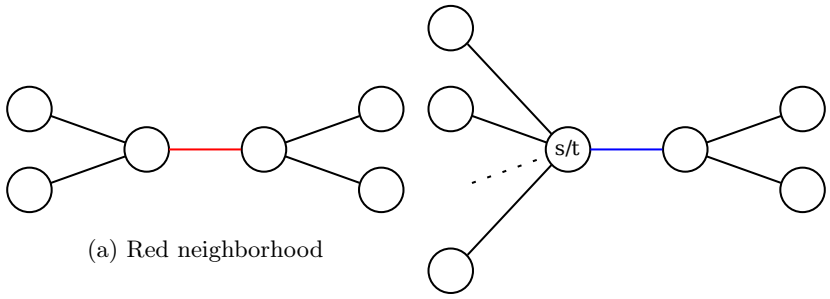


Figure 2: The unweighted graph G_2. The idea is still the same as for the simpler example from fig. 1 in the main paper. But the short parallel paths between s and t have now been replaced by "bands", our name for the subgraphs in colored boxes. This construction ensures that it is impossible to find "safe" edges for contraction based only on local properties. The bands are only hinted at here, the full bands are shown in fig. 1. Blue corresponds to type A, red to B, yellow to C.



(a) Red neighborhood

(b) Blue neighborhood

Figure 3: The different neighborhoods that occur in G_n. s-blue and t-blue are both shown in fig. 3b because they differ only in whether they contain the node s or t.

others are not. A local contraction algorithm assigns the same score to each of these edges with isomorphic neighborhoods. This will give us a lower bound on the contraction probability of edges included in the s-t-mincut cut set in any particular step.

All of the edges in G_n belong to one of three isomorphism classes of neighborhoods, which we call *red*, *s-blue* and *t-blue*. These neighborhoods are shown in fig. 3.

The blue neighborhood classes depend on the degree of s (t) which is a function of n. We will call an edge s-blue (t-blue) if its neighborhood fits the schema from fig. 3b, no matter the degree of s (t). In a fixed graph, all s-blue (t-blue) edges have isomorphic neighborhoods. That the same is not true across graphs does not matter for our purposes.

All the edges are colored according to their neighborhood (blue and red) in fig. 1. During the contraction process, other neighborhoods may of course arise.

The unique s-t-mincut of G_n cuts the red edge in the middle of each band of type A and the blue edges on the side where there is only one of them in bands of type B and C. The contraction algorithm will find this minimum cut iff it does not contract any of these edges. So we will say a contraction is "wrong" or a "mistake" if it contracts one of these edges belonging to the s-t-mincut.

We will now prove some useful statements:

1. In an untouched band, all edges are either red, s-blue or t-blue, with each edge belonging to the same type as in the original graph G_n.

   Proof. It's clear that contractions of red edges in one band don't influence other bands. If a blue edge is contracted, this can change the degree of s or t but has no influence on other bands apart from that. ■

2. If at most $\frac{n}{2}$ bands have been touched, then the probability that contracting a blue or red edge is wrong is $p(\text{wrong}|\text{red or blue}) \geq \frac{1}{118}$.

Proof. There are still at least $\frac{n}{2}$ untouched bands of all three types. Each type contains a red edge, an s-blue edge or a t-blue edge that mustn't be contracted respectively (as per the statement proven just above). So there are at least $\frac{n}{2}$ wrong red edges, $\frac{n}{2}$ wrong s-blue edges and $\frac{n}{2}$ wrong t-blue edges.

Because all red edges have the same neighborhood, the local contraction algorithm assigns the same score to all red edges. The same is true for s-blue edges and for t-blue edges. So we have

$$p(\text{wrong}|\text{red}) = \frac{\#\text{wrong red edges}}{\#\text{red edges}} \geq \frac{\#\text{wrong red edges}}{\#\text{total edges}} \geq \frac{n/2}{59n} = \frac{1}{118}$$

and similarly $p(\text{wrong}|s\text{-blue}) \geq \frac{1}{118}$ and the same for t. This means that $p(\text{wrong}|\text{red or blue})$ is bounded by

$$\begin{aligned} p(\text{wrong}|\text{r or b}) &= \frac{p(\text{wrong}|\text{r})p(\text{r}) + p(\text{wrong}|s\text{-b})p(s\text{-b}) + p(\text{wrong}|t\text{-b})p(t\text{-b})}{p(\text{r}) + p(s\text{-b}) + p(t\text{-b})} \\ &\geq \frac{\frac{1}{118} \cdot p(\text{r}) + \frac{1}{118} \cdot p(s\text{-b}) + \frac{1}{118} \cdot p(t\text{-b})}{p(\text{r}) + p(s\text{-b}) + p(t\text{-b})} \\ &= \frac{1}{118} \end{aligned}$$

$p(r)$, $p(s\text{-b})$ and $p(t\text{-b})$ are what can be influenced by the choice of the scoring function $\mathcal{W}$ but these terms cancel as we can see. ■

We will now prove inductively that contracting an edge in k different bands without contracting any wrong edges happens with probability $\leq \left(\frac{117}{118}\right)^k$ for $k \leq \frac{n}{2}$:

$$p_k := p\left(\text{no mistakes} \mid k \text{ bands touched}\right) \leq \left(\frac{117}{118}\right)^k$$

Proof. $k = 0$ Nothing to show.

$k \to k+1$ The probability of not making any mistakes until $k+1$ bands have been touched is the probability p_k of correctly touching the first k bands times the probability of not making a mistake while touching the final band.

From statement 1 proven above, we know that to touch a new band, a red, s-blue or t-blue edge will have to be contracted at some point. From statement 2 we know that the probability of making a mistake on that single contraction is at least $\frac{1}{118}$. Additional contractions may be made, but they cannot decrease the total probability of making any mistake. So

$$p_{k+1} \leq \left(\frac{117}{118}\right)^k \cdot \frac{117}{118} = \left(\frac{117}{118}\right)^{k+1}$$

which proves the claim for $k+1$.

■

Since all bands have to be touched eventually, we can apply this statement with $k = \frac{n}{2}$. So the success probability is at most $\left(\frac{117}{118}\right)^{n/2}$ which is exponentially low in the number of vertices, $36n + 2$, as claimed. □

Proof of corollary 4. Since the s-t-mincut has cost $3n$, an α-minimal cut may be worse than the mincut by at most $3n(\alpha - 1)$. Every wrong contraction in an untouched band increases the cost of the best cut that is still possible by at least 1 (because there is only one unique way to optimally cut each band). So to find an α-minimal s-t-cut, at most $3n(\alpha - 1)$ wrong contractions may be made in untouched bands.

As there are $3n$ bands, at least $3n - 3n(\alpha - 1) = (6 - 3\alpha)n$ contractions in different bands must be made without mistakes.

We showed in the proof of theorem 3 that making contractions in k different bands without mistakes (with $k \leq \frac{n}{2}$) happens with probability $\leq \left(\frac{117}{118}\right)^k$. If $\alpha < 2$, then $6 - 3\alpha > 0$, and therefore we can apply this result[2] with $k = (6 - 3\alpha)n$ and see that the probability of correctly contracting edges in the required number of bands is exponentially low in n.

Therefore, the probability that mistakes are made in only $3n(\alpha - 1)$ bands is exponentially low, and thus also the probability of finding an α-minimal s-t-cut. □

Proof of theorem 3 for normalized cuts. It suffices to show that the s-t-mincut in G_n is also the normalized cut for large n. The s-t-mincut cuts $3n$ edges. Because the partitions are perfectly balanced in terms of internal edge weights, only cuts that cut fewer edges than that can have a lower normalized cut cost. In particular, any such cut could not separate s and t. One of its partitions could therefore be no larger than one of the bands between s and t. But the normalized cut cost of such a cut approaches 1 for large n, whereas the ncut cost of the s-t-mincut is always $2 \cdot \frac{3n}{2 \cdot 28n + 3n} = \frac{6n}{59n} = \frac{6}{59}$[3]. So the s-t-mincut is indeed also the normalized cut for large n. □

C Implementation of the seeded contraction algorithm

For unweighted graphs, Karger's algorithm can be implemented in $\mathcal{O}(m)$ time as follows [1]: First, a random permutation of all m edges is generated which takes $\mathcal{O}(m)$ time. Afterwards, edges are contracted in the chosen order until only two vertices remain. If an edge has already been removed by previous contractions, it is skipped. [1] also describes how this method can be generalized to weighted graphs. The only change is in how to generate the permutation of edges to give different probabilities to different permutations.

To keep track of when to stop and of the current segmentation at each step, we use a union-find data structure. Keeping this structure up to date increases

[2] If $(6 - 3\alpha)n > \frac{n}{2}$, we just use $k = \frac{n}{2}$

[3] Each partition has $28n$ internal edges and there are $3n$ edges in the cut between partitions

the runtime to $\alpha(n)\mathcal{O}(m)$ [2] where n is the number of vertices and $\alpha(n)$ the inverse Ackermann function. But since $\alpha(n) < 5$ for all practical values of n, this theoretical increase has no practical relevance.

Two modifications are necessary to adapt this implementation to the seeded contraction algorithm: first, we initialize the union-find data structure such that nodes with the same seed are in the same cluster from the beginning. This is possible with a linear scan over all nodes in $\mathcal{O}(n)$.

Second, we keep a boolean vertex property updated that denotes whether a node is already labeled (i.e. in the cluster of a seed node) or not. Whenever we come to an edge connecting two nodes that are already labelled, we skip it instead of merging these nodes. This ensures that no seeds with different labels end up in the same cluster and each node has a well-defined label at the end. These extensions do not increase the runtime of processing one edge beyond $\mathcal{O}(1)$, so the total runtime of the algorithm stays $\mathcal{O}(m\alpha(n))$.

D Details on experiments

D.1 Metrics

We used three common metrics to evaluate performance in the Grabcut experiment: the adjusted Rand index (ARI), variation of information (VoI) and accuracy.

The (unadjusted) Rand index is defined as the accuracy on the space of pairs of samples, in the following sense: count the number TP of pairs of samples that are correctly put in the same cluster (*true positives*) and the number TN of pairs of samples that are correctly put in different clusters (*true negatives*). Then the Rand index is $\frac{TP+TN}{\binom{n}{2}}$ for n samples, where $\binom{n}{2}$ is the total number of pairs of samples.

The adjusted Rand index renormalizes the Rand index such that it is 1 for a perfect clustering and has expected value 0 for a random clustering, independently of the number of clusters. This is done by correcting for chance with

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{1 - \mathbb{E}[\text{RI}]}$$

where RI is the Rand index and the expectation is over permutations of the assigned labels. The expected value of 0 makes the ARI easy to interpret compared to the unadjusted Rand index, for which even random clusterings can have a positive expected value.

The variation of information between two variables X and Y can be defined as

$$\text{VI}(X;Y) := H(X|Y) + H(Y|X)$$

where $H(X|Y)$ is the conditional Shannon entropy. To get a clustering metric, we let X and Y be the different labelings (in our case ground truth and the labeling to be evaluated). The joint distribution over X and Y is defined by picking samples uniformly at random.

D.2 Parameters

In both experiments, we optimized the β parameter by hand to maximize the performance according to the metrics we used separately for each method. In the Grabcut experiment multiple metrics were used, but they all reached their maximum at the same β value of those we tested.

In both experiments, we first found a reasonable range of β values and then tested ten different values within these ranges. For the Grabcut experiment, this range was $\beta = 5$ to $\beta = 100$, for the USPS experiment it was $\beta = 1$ to $\beta = 20$.

D.3 Empirical runtimes

For a complete run on both datasets (Grabcut and USPS), the new Karger-based algorithm takes about 20 minutes[4], the random walker takes about 5 minutes and watershed 9 minutes. All of these are wall clock times when running on 6 CPU cores. The random walker implementation is the one used in SciPy, the other algorithms were implemented by us in Julia. Their runtimes could probably be decreased with a more efficient implementation.

References

[1] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, July 1996.

[2] Robert E. Tarjan and Jan van Leeuwen. Worst-case Analysis of Set Union Algorithms. *Journal of the ACM*, 31(2):245–281, Mar. 1984.

[4] with 100 runs, enough for a reasonably good approximation of the true potential